IaDRA-SCN: Intra-Domain Routing Architecture for Service-Centric Networking

Gasparyan, Mikael ; Braun, Torsten ; Schiller, Eryk

Abstract: Service-Centric Networking (SCN) is a future Internet paradigm derived from Information-Centric Networking (ICN). SCN extends the ICN paradigm with service support. In SCN, services are the key component of the network. In this paper, we design a novel intra-domain routing architecture by integrating an extended version of the Named-data Link State Routing (NLSR) protocol into SCN. We make use of NLSR to disseminate service provider prefixes and resource availability information within the intra-domain network. The resource availability information allows nodes in the intra-domain network to label faces for future forwarding decisions. Our design supports short and long types of service input. We have implemented and evaluated our design against existing schemes through simulations.

DOI: https://doi.org/10.1109/ICCW.2018.8403718

IaDRA-SCN: Intra-domain routing architecture for Service-Centric Networking

Mikael Gasparyan, Torsten Braun, Eryk Schiller
Institute of Computer Science
University of Bern
Bern, Switzerland
{gasparyan,braun,schiller}@inf.unibe.ch

***Abstract*—Service-Centric Networking (SCN) is a future Internet paradigm derived from Information-Centric Networking (ICN). SCN extends the ICN paradigm with service support. In SCN, services are the key component of the network. In this paper, we design a novel intra-domain routing architecture by integrating an extended version of the Named-data Link State Routing (NLSR) protocol into SCN. We make use of NLSR to disseminate service provider prefixes and resource availability information within the intra-domain network. The resource availability information allows nodes in the intra-domain network to label faces for future forwarding decisions. Our design supports short and long types of service input. We have implemented and evaluated our design against existing schemes through simulations.**

Keywords—information-centric networking; service-centric networking; link-state routing; ndn; load balancing

I. INTRODUCTION

The current Internet is host-based meaning that a content consumer needs to possess the provider address to retrieve content. Information-Centric Networking (ICN) aims to replace the host-centric architecture of the Internet by an information-centric one. The fundamental idea behind ICN is to enable a consumer to request a given content object in the network without the knowledge of the content location. To accomplish this goal, ICN has two main types of messages, i.e., Interest and Data messages. The Interest message contains the requested content identifier and is sent by the content consumer. The Data message is the content reply to the Interest message. Message forwarding is based upon unique content identifiers, i.e., ICN centers around content and makes it addressable. There are many ICN implementations that follow the ICN characteristics. Content-Centric Networking (CCN) [1] was one of the first and most influential ICN implementations. Named Data Networking (NDN) [2] has its roots in the CCNx [3] project.

Since more than a decade, the Internet evolution is gradually moving towards a service-oriented design. Consequently, Future Internet architectures should also consider this development by providing service support. Service-Centric Networking [4] is an evolution of ICN by generalizing its content-oriented design towards a service-oriented scheme. In contrast to simple content retrieval, services often require some level of processing at the service provider. Consequently, load balancing is one of the critical aspects. The workload originated from consumers requesting processing needs to be evenly distributed. Service input parameter support is another essential requirement for service requests. Our work focuses on input parameter support and load balancing for service requests in SCN. In this paper, we present a novel SCN architecture and a load balancing mechanism based on NDN and NLSR (Named-data Link State Routing) [5]. Our design includes innovative support for service requests requiring short (i.e., small variables) and long (i.e., NDN objects or content stored locally on the service consumers) input parameters. We enhance the NDN architecture by integrating service request and service input parameter support. Therefore, NDN primitives can be used to request content and services. We implement and integrate an extended version of NLSR, which is capable of propagating service resource availability information in the network.

This paper is organized in the following way. Related work is discussed in Section II. Our architecture called IaDRA-SCN (Intra-domain routing architecture for Service-Centric Networking) is presented in Section III. Section IV evaluates IaDRA-SCN. In Section V, we conclude this paper.

II. RELATED WORK

SCN is an extension of the ICN paradigm, it leverages ICN with service support. There is a substantial number of work that tackles ICN and SCN challenges. To the best of our knowledge, however, there are only a few works that focus on the architectural design of SCN. This section will give a brief overview of existing architectures and mechanisms related to load balancing and support for input parameters in SCN. There is a considerable number of service-oriented approaches managing load with the help of a centralized coordinator. The coordinator has a global topology overview, and it also possesses knowledge of available resources in service providers. Using the global view of the network and the available resource information, the coordinator can provide load balancing among service providers. Mechanisms based on the centralized paradigm have a single point of failure, which should be avoided [6].

Serval [7] modifies the TCP/IP stack by integrating a new layer named Service Access Layer (SAL). SAL enables applications to communicate with one another by using service names. SAL sits above the network layer and uses tables to map service names into network addresses, permitting service name-based forwarding. Serval provides service request load

balancing and service maintenance capabilities. The Serval architecture leverages special service routers, which are responsible for locating available service replica that satisfy an incoming request.

CCNxServ [8] is an extension of the ICN architecture built on top of CCNx [3]. CCNxServ allows dynamic deployment of services in the network. Prior to service request processing, a node fetches the required service application from the network. CCNxServ integrates the NetServ [9] virtualization framework for services. However, the NetServ component incorporated into CCNxServ is IP-based, which contradicts the principles of ICN. CCNxServ does not provide any load balancing mechanism.

NextServe [10] leverages CCN by integrating service support. NextServe uses a service composition and a human-readable naming scheme inspired by object-oriented programming. NextServe is similar to CCNxServ. A service consumer must fetch the required service executable from the network before executing processing. NextServe does not integrate load balancing capabilities either.

The Service over Content-Centric Routing (SoCCeR [11]) architecture integrates service support into CCN. SoCCeR leverages CCN with Ant Colony Optimization [12] to rank outgoing faces for future forwarding decisions. SoCCeR adds a layer in the CCN architecture, which aims to periodically broadcast Ant Interest messages for a given service that is available in the network. Ant broadcast messages traverse the whole network until they reach a node providing the requested service. Such a node will then reply to the Ant Interest with an Ant Data message that contains information about resource availability of the service provider nodes. The information about a given service and its availability gathered in the Ant Data reply messages are used by the intermediate nodes to rank outgoing faces for future forwarding. SoCCeR does not support input parameters for service requests. There is no freely available implementation of SoCCeR.

Named Function Networking (NFN) [13] extends the CCN framework by integrating service support. NFN adds two additional layers called NFN Layer and Service Layer on top of CCNx [3]. The NFN layer incorporates a lambda expression engine and is responsible for forwarding decisions. The service layer manipulates the data by running the NFN engine. In NFN, a service request holds the data block and a collection of functions that must be executed over the data. If a service provider is overloaded by service requests, computing can be redirected to the neighboring nodes. NFN does not provide local input parameter support.

In our previous work, we have created L-SCN (Layered SCN architecture with supernodes and Bloom filters) [14], which is a two-layered SCN architecture and splits the network into domains. It uses Bloom filters and supernodes to propagate information about services and available resources of service providers through the network. The Bloom filter dissemination mechanism enables L-SCN to propagate information about momentarily available service prefixes in an efficient manner (i.e., low protocol overhead). L-SCN supports short and long input parameters. However, it does not support the use of regular content Interest name as input parameter. Moreover, its input parameter support scheme is push-based, the service consumer pushes its input parameters to the service provider trough Interest messages, this contradicts the principle of the NDN paradigm.

III. IADRA-SCN

This paper presents our proposed intra-domain routing architecture named IaDRA-SCN (Intra-domain routing architecture for SCN), which takes advantage of the NLSR routing information dissemination scheme. IaDRA-SCN integrates our support scheme for short and long (i.e., NDN objects or locally stored content) types of input parameters. In contrast to existing solutions, our design is based upon a link-state routing protocol adapted for Service-Centric Networking. IaDRA-SCN uses NLSR, which does not use flooding. Instead, it uses a hop-to-hop synchronization mechanism based on ChronoSync [15]. Prefixes, Adjacency, and Resource Availability (e.g., CPU, RAM) for each node are disseminated with Link State Advertisements (LSAs). This propagation allows each node in the network to build the network topology and to identify nodes associated with prefixes. Additionally, the dissemination of resource availability allows us to rank outgoing faces for subsequent service request forwarding decisions.

A. NDN Extensions

In our work, we use the NDN implementation of ICN. NDN is based on a fork of CCNx. NDN has two types of messages called Interest and Data. The content requester sends Interest messages carrying the name that identifies a given content object. A content name is composed of various components separated by a slash character (e.g., /myrep/video/file1.mpg). A Data message is a reply to the Interest message and contains the identifier and the requested content. NDN has three main routing components that provide the core of its routing architecture: Forwarding Information Base (FIB), Pending Interest Table (PIT), and Content Store (CS). The FIB is a data structure that stores a set of forwarding faces for content identifiers. The purpose of the FIB is similar to the routing table in IP networks. However, routing is based on content names instead of addresses. The PIT stores all incoming Interest requests that have been forwarded further, but have not been satisfied yet with a content reply. Incoming Data messages are cached in the CS. CS allows for a quick response to subsequent incoming Interest requests directly from the local cache. We extend NDN to support also service requests by distinguishing regular NDN content requests and service requests. A service Interest name starts with prefix *service* and specifies that a given Interest is a service request. The prefix is followed by the service identifier (e.g., /service/service1). To forward incoming Interest requests, the FIB needs to be populated with the name and the face information.

B. Parameter Support for Service Requests

In Sec. III-A, we argue that requests for content and services should use distinct prefixes. In this section, we present different methods providing parameters to services. As we demonstrate, input parameters become essential components of

service requests. Our mechanism handles short and long types of input parameters for service requests.

We assume that a short parameter does not exceed 1 KB. Please notice that the NDN naming convention allows us to include short parameters in the NDN Interest name. However, in general NDN is not designed to push large amount of content with Interest messages from the consumer towards the producer. We also believe that pushing long objects contradicts the core principle of NDN meaning that the content has to be pulled. Therefore, long input parameters should be fetched by the service provider. We aim to integrate a parameter handling scheme that natively handles long parameters (i.e., massive amounts of data). As described in section III-A, a typical service request should contain a service prefix (/service) followed by the service identifier. We further extend the Interest name of the service request with additional pieces of information about the service input parameters. Short parameters can be directly provided within Interest names. For long parameters, we, however, only provide the information (i.e., a pointer) allowing the service provider to fetch the long parameter from the network. Please notice that the long parameter can target a public object residing in the NDN network or a private resource on the service consumer, i.e., not registered in the NDN network. A long parameter identified has to be ordinarily fetched by the service provider as regular content from the NDN network. If the content remains private, the service provider has to directly contact the service consumer and request a given parameter. The subsequent paragraphs describe our design of input parameters.

Each input parameter uses a combination of two components of the NDN naming convention. The first component indicates the type of the input parameter and its position in the set of parameters. The type could be either short (i.e., s) or long (i.e., l). For example, s1 indicates that the first input parameter is of type short, l2 indicates that the second parameter is of type long. The role of the second component varies depending on the parameter type. In case of a short parameter (i.e., s), the second component carries the content of the parameter. In case of a long parameter, the second component is a pointer towards the parameter, we call it an input parameter identifier. The input parameter identifier can be a content Interest name or an Interest name containing the result of a hash (e.g., MD5, SHA) computed over the content. When the service consumer uses an Interest name as an input parameter identifier, the service provider is able to fetch the content of the requested input parameter from the NDN network using a regular Interest for content. When the content of the input parameter is private to the consumer (not registered in the NDN network) meaning that there is no prefix available for such content in the FIBs, we implement a special parameter retrieval procedure. In this case, the service consumer provides the hash of the content. With very high probability, the content can be uniquely distinguished through the hash. The service provider must in turn use a special Interest (containing the hash of the parameter) to fetch the parameter from the consumer. The Interest message, the Data message, and the forwarding procedure in the retrieval of the long parameter will be explained in subsequent paragraphs.

To illustrate the naming convention for input parameters, let us consider the following example of a service using two parameters. The first parameter is a short value, equal to an integer (e.g., 7). The second is a long parameter (e.g., 0cc175b9c0f1b6a831c399e269772661) content stored privately at the consumer. Such content does not have a prefix registered in the network. The component set appended to the NDN service request providing input parameters is equal in this example to /s1/7/l2/0cc175b9c0f1b6a831c399e269772661. The first two components indicate that the first parameter is of type *s* and has a data value of 7. The second two components describe the second parameter of type long (*l*). The input parameter identifier is a hash computed over the parameter, e.g., a large file that is too big to be provided through the service request Interest name. Therefore, the object needs to be fetched from the consumer by the service provider using a subsequent Interest equipped with the corresponding hash value (0cc175b9c0f1b6a831c399e269772661 in our example). On the other hand, the input parameter identifier can also be a regular NDN Interest name. In that case, if an Interest name is given, the input parameter can be retrieved from the NDN network, because the content prefix exists in the network. To conclude, the service request Interest name consists of the following elements: service prefix, service identifier, and service input parameter(s). The service prefix indicates that the request is for a service (not for content). The service identifier specifies the requested service. The remaining part indicates the service input parameters of short and long types.

In general, when a service provider receives a service request, it extracts the short type parameter values from the Interest name and fetches the long type parameter content by using the provided long parameter hashes or content names. Let us consider an example in Table I, which presents a final service request. The service name is *zip*. The zip service request has a single input parameter of type long identified through a hash. The parameter identifier is a hash equal to 4124bc0a9335c27f086f24ba207a4912. The service provider must then request the content from a consumer using a special content Interest. The following paragraph explains the procedure of retrieving long parameters identified through a hash by using a special content Interest. The procedure of retrieving long objects identified through a content identifier will not be thoroughly explained as it is a regular NDN content retrieval explained in [2]. For a regular NDN content, the service provider sends an Interest with the extracted content name received from the consumer.

TABLE I. COMPONENTS OF A SERVICE REQUEST

Components of a service request with one input parameter	
Name	***Complete service request with one long hash identifier parameter***
Service prefix	/**service**/zip/l1/4124bc0a9335c27f086f24ba207a4912
Service identifier	/service/**zip**/l1/4124bc0a9335c27f086f24ba207a4912
Service input parameter	/service/zip/**l1/4124bc0a9335c27f086f24ba207a4912**

However, when the parameter is private to the consumer and its corresponding content name is not registered in the network (i.e., no FIB entries), the service provider uses the hash received from the consumer and sends an Interest name to fetch the content: /HASH, in which *HASH* is the received hash from the consumer. Let us consider the example in Table I. Upon receiving the service request, the service provider realizes that it needs to fetch the parameter from the service consumer by using the *HASH* 4124bc0a9335c27f086f24ba207a4912. Therefore, the service provider issues a data parameter request using the following Interest name: /4124bc0a9335c27f086f24ba207a4912. Typically, in NDN, regular content and service requests are forwarded using FIB entries. Therefore, the intermediate nodes need to update their FIB tables with a new entry that allows message forwarding for such special Interest names. This allows the Interest containing the hash sent by the provider to reach the service parameter content at the service consumer. The process of updating FIB tables on the forwarding nodes is described in the following paragraph. Security and privacy concerns are beyond the scope of this paper. However, a public-key encryption mechanism can be introduced between service producer and consumer for secure communication, e.g., keep private objects protected from third parties.

Fig. 1 illustrates the service request containing an input parameter identifier issued by the service consumer C. Please consider the form of the service request described in Table I. The service request is forwarded to N2 and N3 until it arrives at service Provider P3. The forwarding uses the NDN forwarding scheme, which forwards the Interest according to a selected NDN forwarding strategy. Upon the service request forwarding process, the intermediate nodes verify whether the service request contains hash identifiers for input parameters. In the case hash identifiers are discovered, forwarding nodes alter the necessary FIB tables. In our example, the forwarding of the service request /service/zip/i1/4124bc0a9335c27f086f24ba207a4912, requires the intermediate nodes to update their FIB table with a new entry: /4124bc0a9335c27f086f24ba207a491, that leads towards the service consumer. Such an entry allows subsequent Interest messages for a given input parameter identified by a hash to be forwarded from the service provider towards the consumer through appropriate faces. When the service request sent by consumer C passes through intermediate nodes N2 and N3 (c.f., Fig 1), N2 and N3 update their FIB tables by adding corresponding entries for the input parameter name (hash). When the request arrives at provider P3, P3 extracts the input

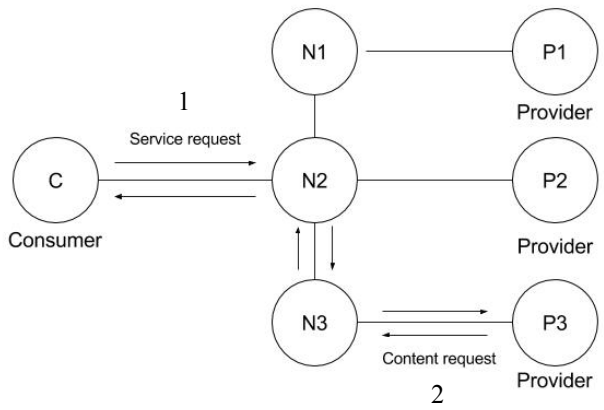

Fig. 1. Service request send by service Consumer C followed by the input parameter retrieval request of service Provider P3.

parameters from the Interest name and sends an Interest for each input parameter identifier. In our example, the service provider sends an Interest message to fetch the content identified by /4124bc0a9335c27f086f24ba207a4912 from the consumer. N3 and N2 can forward the Interest to Consumer C, because they constructed appropriate FIB entries upon service request forwarding. It is worth noting that an Interest message containing an input parameter uses the same underlying forwarding mechanisms as regular content Interest messages. Consequently, it benefits from NDN capabilities such as caching. This allows content of input parameters to be equally cacheable in the network as regular content. Multiple service requests requiring the same input parameters may be effectively handled, because input parameters are considered regular content distinguished by a unique naming prefix.

C. Service Provider Information Propagation with NLSR

This subsection tackles the dissemination of information about services in the network. NLSR is a routing protocol that populates FIB tables of NDN nodes. NLSR uses Interest and Data packets to disseminate routing information. It discovers adjacencies and advertises information about the topology and name prefixes available in the network. For this purpose, NLSR disseminates two Link State Advertisements (LSAs) – Adjacency LSA and Prefix LSA. The Adjacency LSA propagates all active links connecting a node to its neighbors. The Prefix LSA is used to propagate information about existing prefixes registered on each node. We add an additional LSA, which periodically propagates node resource availability information. LSA messages are not propagated by a host-centric push mechanism, but through regular NDN Interest and Data messages. LSAs are gathered by LSA Interest messages, which receive a Data reply containing the LSA content. NLSR uses ChronoSync [15] as synchronization protocol. ChronoSync uses cryptographic digests to summarize the state of a dataset in a condensed way. The digest is exchanged among network parties to detect differences in the datasets and to disseminate it to the network efficiently. NLSR uses a Hello protocol, which allows detecting a node failure by sending periodic Interest messages to the neighbors. A Data message reply to the Interest message informs us that the neighboring node is alive. As a link-state protocol, NLSR disseminates Link State Advertisements (LSAs) to build a network topology and to distribute name prefix reachability. We extend the LSAs by adding the ability to propagate resource availability periodically. The resource availability dissemination allows efficient load balancing for service requests. This is important, because service requests often require substantial processing.

Nodes need knowledge about available services and resources in the network to forward service request to the service provider with adequate available resource. A node needs information on faces about service reachability and available resources at service providers. Service providers propagate the service and resource information to the network. To enable the propagation of routing information in the network, our design makes use of LSAs [5]. NLSR uses NDN Interest and Data packets to ensure propagation of routing information. It uses LSAs to propagate information and a Link State Database (LSDB) on every node to store the latest

version of LSAs. It uses a specific naming convention for information dissemination among routers, which is of the following structure: /{network}/NLSR/LSA/{site}/{router}/{lsa-type}/{version}. The components of the naming scheme are described in Table II.

TABLE II. LSA NAMING FORMAT

Component	*Description*
{network}	the network name to which the router belongs
NLSR	indicates that it is an NLSR request
LSA	indicates that it is an LSA
{site}	the site name to which the router belongs
{router}	the router that originates the LSA
{lsa-type}	the type of the LSA
{version}	version of the LSA incremented each time a new LSA is built

The latest version of the LSAs are stored on each router in a database called LSDB. NLSR uses the ChronoSync [15] protocol to synchronize LSDB changes in the network. Information propagation is realized through content synchronization among routers. The ChronoSync keeps LSAs in the LSDB as a name set. To synchronize the latest version of the LSAs, the ChronoSync protocol exchanges a hash as a compact expression of the LSA name set. Using the hash exchange, ChronoSync avoids unnecessary flooding of the network. Instead of requesting LSA names by flooding the network, NLSR first checks if there are new LSAs available by exchanging a hash of the locally available LSA set. If the received hash is different (i.e., the LSAs were updated), NLSR starts requesting new LSAs.

The service providers need to disseminate information about the services provided and the available resources to the network. For this purpose, service providers propagate Service Prefix Information and Resource Availability Information LSAs to the network, LSA dissemination mechanism is described in detail in [15]. The Service Prefix Information LSA is advertised each time a new service (identifier prefix) is added or deleted by a service provider. The Service Prefix Information LSA is used to update the FIB routing tables of the nodes for future forwarding decisions. The Resource Information LSA periodically updates the available resource information (e.g., CPU, GPU, RAM) of a given service node. The nodes use the resource availability information for future forwarding decisions. It allows forwarding requests to the provider with the currently most available resources. The design respects the NDN primitives because information is solicited with Interest messages.

IV. EVALUATION AND RESULTS

This section presents an evaluation of our IaDRA-SCN architecture. We compared IaDRA-SCN with existing forwarding strategies in NDN, which are the Best Route, Multicast, and Random strategies [16]. The Best Route strategy forwards incoming requests to the lowest cost provider node by using networking inspired metric values. The Multicast strategy forwards an incoming request to multiple faces, through which the requested Interest is accessible. The Random strategy forwards the request through a randomly chosen face from the set of faces that can lead to the requested service.

A. Evaluation Scenario

We have implemented and evaluated IaDRA-SCN in the ndnSIM [16] simulator. NdnSIM is an NDN simulator for ns-3. NdnSIM does not use DCE (direct code execution). Nevertheless, our code needs only a few changes to become a real deployable system. We have modified NDN to provide service request forwarding. We extended the message and forwarding scheme. For service information propagation, we have integrated and extended NLSR with service information dissemination as described in Sec. III-C. The underlying architecture of NDN was left unchanged. We only extend it with additional NDN-native mechanisms to support services. Thus, regular NDN traffic is handled as usual.

We evaluate our IaDRA-SCN architecture using an example topology (c.f., Fig. 2) generated by the Internet topology generator BRITE [17]. We use the flat Albert-Barabasi model to generate a router level topology. The created topology is composed of 50 nodes with a minimum node degree of 1. In BRITE, we use preferential connectivity and incremental growth setting allowing us to produce a topology that highly resembles the real-world Internet. We have randomly designated 4 service provider nodes and 6 service consumer nodes. The LSA refresh time interval is set to 5 seconds. Service consumer nodes send service requests using values drawn from a random exponential function with a mean of 2 seconds. The service provider nodes aim to process incoming service requests sent by the service consumers. The processing time of incoming service requests is set to a uniformly distributed value between 1 and 2 seconds. This implies that the resources on the node are busy during the service processing time. In the defined scenario, each service consumer node sends 100 unique service requests. That sums up to 600 unique service requests that need to be processed by the service provider in the network. The processing delay is defined as the time elapsed at the service consumer between a request origination and the result arrival. The simulation was repeated 10 times for each strategy considered. Finally, we have calculated the mean processing delay and the 95% confidence intervals for the mean.

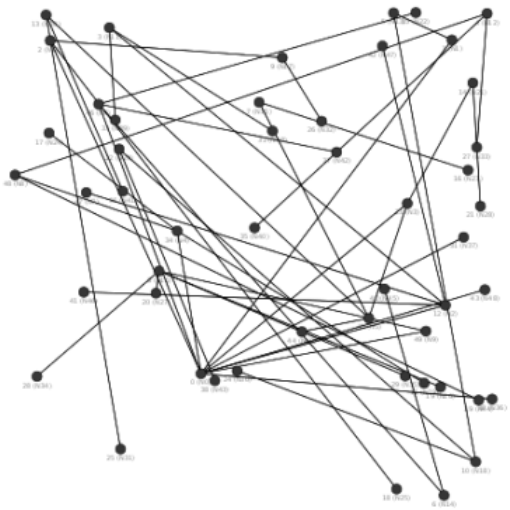

Fig. 2. Evaluation topology composed of 50 nodes

B. Results

We compare the processing delay of IaDRA-SCN with the Best Route, Multicast, and Random forwarding strategies existing in ndnSIM [16] with default settings. Other mechanisms were not considered, because their implementations were not available. The mean processing delays for all strategies considered are shown in Fig. 3. The circles and bars represent the mean processing delays, and the ranges express the confidence intervals for a given strategy. The confidence levels for the strategies considered do not intersect at all, suggesting significant differences between the mean processing delays of the four strategies considered.

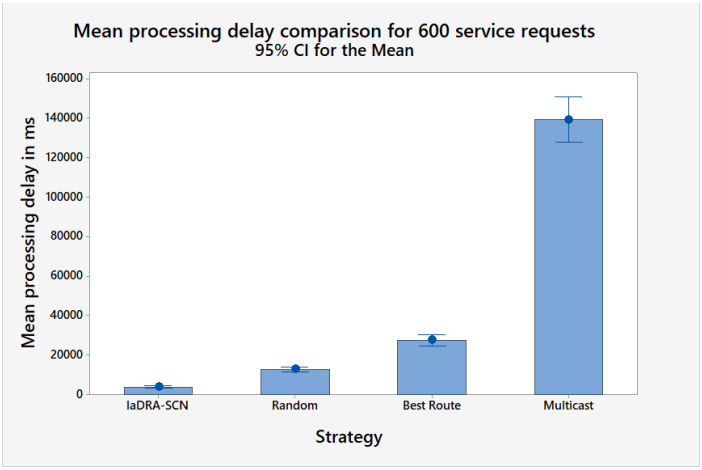

Fig. 3. Mean processing delays and confidence intervals for the mean for 600 service requests.

The Multicast strategy reveals the highest mean processing delay. This is because each service request is forwarded to multiple nodes. Therefore, this strategy quickly saturates computing resources resulting in huge processing delays. The Best Route strategy relies on network-based metric values (e.g., hop count and the number of previously forwarded service requests) to select the outgoing forwarding face. The use of network metrics overloads best reachable providers, thus increasing their load and effectively slowing down the service. Best Route provides a mean processing delay of 27709 (±3011) ms. The Random achieved the best load balancing performance with a mean processing delay of 12916 (±1089) ms. IaDRA-SCN, however, significantly outperforms the three strategies with a mean processing delay of 3974 (±621) ms. The evaluation shows that IaDRA-SCN produces outstanding results compared to the strategies in ndnSIM. This is due to the integration of an efficient load balancing scheme based on information propagation with the link-state routing protocol.

V. Conclusion and Future Work

According to our knowledge, IaDRA-SCN is the first intra-domain SCN architecture using a link-state routing protocol and integrating short and long type input parameter support. IaDRA-SCN disseminates information about link connectivity, available services, and resources to the entire network. For the propagation of network related information, it uses a synchronization mechanism instead of flooding. The disseminated information allows nodes to maintain a global view of the network. Consequently, it enables us to forward incoming service requests to the service provider with the most available resources. Our design does not push content to the network, hence respects the NDN primitives, which aim to solicit content with Interests. We have implemented a prototype of IaDRA-SCN and compared it against existing forwarding strategies available in ndnSIM. The evaluation results reveal that IaDRA-SCN outperforms existing strategies in terms of mean processing delay. As future work, we aim to assess other aspects of IaDRA-SCN such as the limits of its scalability by considering other performance metrics such as protocol overhead.

References

[1] Jacobson, V., Mosko, M., Smetters, D., & Garcia-Luna-Aceves, J. J. “Content-centric networking”. Palo Alto Research Center, 2007.

[2] "Named Data Networking (NDN) - A Future Internet Architecture", Named Data Networking (NDN), 2017. [Online]. Available: http://named-data.net.

[3] "CCNx | PARC's implementation of content-centric networking", Blogs.parc.com, 2017. [Online]. Available: http://blogs.parc.com/ccnx/.

[4] T. Braun, V. Hilt, M. Hofmann, I. Rimac, M. Steiner and M. Varvello, "Service-Centric Networking", 2011 IEEE International Conference on Communications Workshops (ICC), 2011.

[5] V. Lehman, A. M. Hoque, Y. Yu, L. Wang, B. Zhang, and L. Zhang, “A secure link state routing protocol for NDN,” NDN, Technical Report NDN-0037, Jan. 2016.

[6] Hussain, Hameed, S. Malik, A. Hameed, S. Khan, et al., "A survey on resource allocation in high performance distributed computing systems", Parallel Computing, vol. 39, no. 11, pp. 709-736, 2013.

[7] Nordström, E., Shue, D., Gopalan, P., Kiefer, R., Arye, M., Ko, S. Y., .& Freedman, M. J. (2012, April). Serval: An end-host stack for service-centric networking. In Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation USENIX Association.

[8] S. Srinivasan, A. Singh, D. Batni, J. Lee, H. Schulzrinne, V. Hilt and G. Kunzmann, "CCNxServ: Dynamic service scalability in information-centric networks", 2012 IEEE International Conference on Communications (ICC), 2012.

[9] S. Srinivasan, J. Lee, E. Liu, M. Kester, H. Schulzrinne, V. Hilt, S. Seetharaman and A. Khan, "NetServ", Proceedings of the 2009 workshop on Re-architecting the internet - ReArch '09, 2009.

[10] D. Mansour, T. Braun and C. Anastasiades, "NextServe Framework: Supporting Services over Content-Centric Networking", Lecture Notes in Computer Science, pp. 189-199, 2014.

[11] Shanbhag et al. "SoCCeR: Services over content-centric routing." Proceedings of the ACM SIGCOMM workshop on Information-centric networking. ACM, 2011.

[12] M. Dorigo and T. Stützle, Ant colony optimization. Cambridge, Mass.: MIT Press, 2004.

[13] C. Tschudin and M. Sifalakis, "Named Functions for Media Delivery Orchestration", 2013 20th International Packet Video Workshop, 2013.

[14] M. Gasparyan, T. Braun and E. Schiller, "L-SCN: Layered SCN architecture with supernodes and Bloom filters", 14th IEEE Annual Consumer Communications & Networking Conference (CCNC), 2017.

[15] Z. Zhu and A. Afanasyev, "Let's ChronoSync: Decentralized dataset state synchronization in Named Data Networking", 2013 21st IEEE International Conference on Network Protocols (ICNP), 2013.

[16] "ndnSIM Documentation — ndnSIM documentation", Ndnsim.net, 2017. [Online]. Available: http://ndnsim.net/.

[17] A. Medina, A. Lakhina, I. Matta and J. Byers, "BRITE: an approach to universal topology generation", MASCOTS 2001, Proceedings Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems.